Small-step semantics

$$e \longrightarrow e'$$

models each 'step' of computation *within* the judgment.

Big-step $e \Downarrow v$:
("evaluation semantics")

$$\dfrac{\dfrac{}{(\text{num } 1) \Downarrow (\text{num } 1)} \quad \dfrac{\dfrac{}{(\text{num } 2) \Downarrow (\text{num } 2)} \quad \dfrac{}{(\text{num } 10) \Downarrow (\text{num } 10)}}{(\text{add } (\text{num } 2)\ (\text{num } 10)) \Downarrow (\text{num } 12)}}{(\text{add } (\text{num } 1)\ (\text{add } (\text{num } 2)\ (\text{num } 10))) \Downarrow (\text{num } 13)}$$

Small-step $e \longrightarrow e'$:

$$\dfrac{\dfrac{}{(\text{add } (\text{num } 2)\ (\text{num } 10)) \longrightarrow (\text{num } 12)}\ \text{step-add}}{(\text{add } (\text{num } 1)\ (\text{add } (\text{num } 2)\ (\text{num } 10))) \longrightarrow (\text{add } (\text{num } 1)\ (\text{num } 12))}\ \text{step-context}$$

$$\dfrac{}{(\text{add } (\text{num } 1)\ (\text{num } 12)) \longrightarrow (\text{num } 13)}$$

Advantages of small-step:

- can model behaviour of nonterminating programs
- closer to what the machine actually does
- can model parallelism $\Longrightarrow$ a4

$e1 \longrightarrow e2 \quad e2 \longrightarrow e3 \quad e3 \longrightarrow e4 \quad e4 \longrightarrow v$

$e1 \Downarrow v$

Small-step semantics: review of evaluation contexts

("context": an overused word)

- Reduction rules describe the 'real' computation:
  - adding,
  - applying a lam,
  - handling (if0 (num ...) ...)

They can only be applied if the 'real' computation is the whole expression:

✓ $\dfrac{}{(\text{add } (\text{num } 2)\ (\text{num } 10)) \longrightarrow (\text{num } 12)}$ Step-add

$\dfrac{}{(\text{add } (\text{num } 1)\ (\text{add } (\text{num } 2)\ (\text{num } 10))) \not\longrightarrow}$ ~~Step-add~~

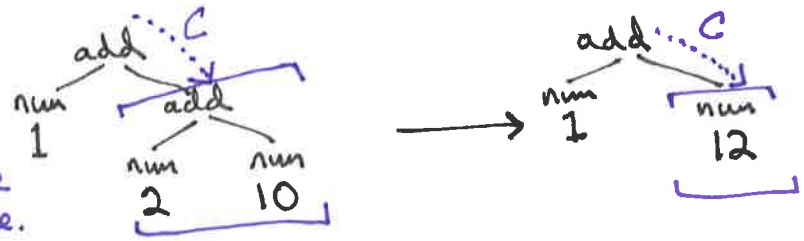
not the whole expression

- The CONTEXT RULE, Step-context, lets us look inside the expression and reduce an inner expression.

$$\dfrac{\dfrac{}{(\text{add } (\text{num } 2)\ (\text{num } 10)) \longrightarrow (\text{num } 12)}\ \text{Step-add}}{(\text{add } (\text{num } 1)\ (\text{add } (\text{num } 2)\ (\text{num } 10))) \longrightarrow (\text{add } (\text{num } 1)\ (\text{num } 12))}\ \text{Step-context}$$

$C[\ e\ ]$ $\qquad$ $C[\ e'\ ]$

The context is like a path from the root of the abstract syntax tree to a subtree.



$C ::= [\,]$ "here"

$\mid$ (add C e) "go to the first subexpression inside add" (subtree)

$\mid$ (add v C) "go to the second subexpression" — if we've evaluated/stepped the first subexpression

$\mid$ ⋮

Small-step semantics for parallelism:

(par e1 e2) : "run e1 and e2 in parallel" and use the result of whichever expression finishes first

$$\left(\frac{e_1 \Downarrow ?}{(\text{par } e_1\ e_2) \Downarrow ?}\right)$$

Reduction rules for par:
"Are we there yet?"

$$\frac{}{(\text{par } v_1\ e_2) \longrightarrow v_1}\ \text{step-par-left} \qquad \frac{}{(\text{par } e_1\ v_2) \longrightarrow v_2}\ \text{step-par-right}$$

Contexts:

$$\begin{aligned} C ::= {} & [\,] \\ \mid {} & (\text{add } C\ e) \\ \mid {} & (\text{add } v\ C) \\ \mid {} & \vdots \\ \mid {} & (\text{par } C\ e) \\ \mid {} & (\text{par } e\ C) \end{aligned}$$

∴ "here"